A METHOD FOR CLASSIFYING MULTISPECTRAL REMOTE SENSING DATA USING CONTEXT

PHILIP H. SWAIN, HOWARD J. SIEGEL, AND BRADLEY W. SMITH

Purdue University

ABSTRACT

A statistical model of spatial context is described and procedures for classifying remote sensing data using a context classifier are outlined. Experimental results are presented. Because the computational requirements of the context classifier are very large, its implementation on parallel/pipelined multiprocessor systems is being investigated. Some of the special considerations necessary for such implementations are described, with particular reference to implementation on an array of Control Data Corporation Flexible Processors.

I. INTRODUCTION

For more than a decade, efforts to extract information from multispectral remote sensing image data have proved increasingly successful. To a large extent, these efforts have focused on the application of pattern recognition techniques to the multispectral measurements made on individual ground resolution elements; i.e., scenes have been classified pixel-by-pixel based on the measurement vectors associated with the individual pixels[1]. Progress has been achieved through development of increasingly sophisticated methods for extracting information from the spectral domain to characterize the classes of interest.

However, there are many applications for which the classes of interest can be better characterized if the spatial information in the remote sensing data is utilized in addition to the spectral measurements. Characteristic spatial features include, for example, shape, texture, and structural relationships.

This work was sponsored in part by the National Aeronautics and Space Administration under Contract NAS9-15466.

Some interesting and useful research has been accomplished in recent years in the direction of incorporating spatial information into the data analysis process[2,3,4].

One way to approach spatial information in image data is to recognize that the ground cover associated with a given pixel, i.e., its "class," is not independent of the classes of its neighboring pixels. Stated in terms of a statistical classification framework, we may have a better chance of correctly classifying a given pixel if we take account of not only the spectral measurements associated with the pixel itself but of the measurements and/or classifications of its "neighbors" as well. Notice that at some point we must make clear how "neighborhood" is to be defined.

If the objects in the scene tend to be rather large relative to the resolution of the sensor, i.e., each object is likely to consist of many spectrally similar pixels, this fact can be exploited nicely by applying a combination of scene segmentation techniques and sample classification (sometimes called "per-field" classification)[3]. More generally, the image can be considered a two-dimensional random process and the characteristics of this process incorporated into the classification strategy. This is the objective of the approach described here, in which a form of compound decision theory is employed to improve scene classification through use of a statistical characterization of context. Our work is an extension of an idea by Welch et. al.[5]

As increasingly complex forms of data and data analysis methods are employed, the computational requirements tend to become more demanding. Although improvement in the raw speed of digital computer components can be exploited to some extent to meet these requirements, it is clear that evolving computer architectures, especially those involving multiple processing elements, have much to offer. The

CH1430-8/79/0000-0343$00.75 © 1979 IEEE

context classifier described here has computational requirements which are severe and become more so as the size of the contextual neighborhood is expanded. It is a natural candidate, therefore, for multiprocessor implementation.

II. THE CONTEXT CLASSIFIER

The image data to be classified is assumed to be a two-dimensional $N_1 \times N_2$ array of multivariate pixels. Associated with the pixel at "row i" and "column j" is the multivariate measurement vector $X_{ij} \in R^n$ and the true state or class of the pixel $\theta_{ij} \in \Omega = \{\omega_1, \ldots, \omega_m\}$. The measurements have class-conditional densities $p(X|\omega_i)$, i = 1, 2, ..., m, and are assumed to be class-conditionally independent. The objective is to classify the $N = N_1 \times N_2$ observations in the array.

The action (classification) determined by the classifier for pixel (i, j) is denoted by $a_{ij} \in \Omega$. To pursue a Bayesian (minimum risk) strategy , let the loss incurred by taking action a_{ij} when the true class is θ_{ij} be denoted by $L(\theta_{ij}, a_{ij})$ for some fixed non-negative function $L(\cdot, \cdot)$. The average loss incurred over the N classifications in the array is defined to be

$$\frac{1}{N} \sum_{i,j} L(\theta_{ij}, a_{ij}). \qquad (1)$$

In the most general case, the action a_{ij} may depend on all of the observations in the array. Let $\underline{X}$ denote this "vector of vectors"; then the expected loss is

$$R(\underline{X}) = E\left[\frac{1}{N} \sum_{i,j} L(\theta_{ij}, a_{ij}(\underline{X}))\right] = \frac{1}{N} \sum_{i,j} E[L(\theta_{ij}, a_{ij}(\underline{X}))] \qquad (2)$$

and we would like to have a decision rule (the rule of choosing a_{ij} based on $\underline{X}$) which minimizes $R(\underline{X})$. Note that the expectation is with respect to θ_{ij}.

When context is ignored, the action (classification) depends only on the measurement vector X_{ij} of the pixel to be classified, in which case $a_{ij}(\underline{X}) = a_{ij}(X_{ij})$. For our present purposes, however, we want to incorporate some neighborhood information in the decision process, so we define a neighborhood, the "context," consisting of an arrangement of p pixels such as shown in Figure 1. The arrangement actually used will be based on physical and other practical considerations related to the environment and application. Let $\underline{X}_{ij}$ be a p-vector of measurement vectors associated with pixel (i, j) to be classified and let $\underline{\theta}_{ij}$ be the corresponding p-vector of actual classes. The function $a_{ij}(\underline{X}_{ij})$ maps p-vectors of observations into single classes (i.e., classifies pixel (i, j) based on $\underline{X}_{ij}$). The expected loss over the full array is

$$R(\underline{X}) = \frac{1}{N} \sum_{i,j} E[L(\theta_{ij}, a_{ij}(\underline{X}_{ij}))] \qquad (3)$$

Furthermore, if $L(\cdot, \cdot)$ is taken to be the 0-1 loss function (no loss for a correct classification, unit loss for an error) and the measurements in a neighborhood are assumed to be class-conditionally independent, it can be shown that Eq. (3) will be minimized if every pixel is classified (action a is selected) so as to maximize

$$\sum_{\substack{\underline{\theta}_{ij} \in \Omega^p, \\ \theta_{ij} = a}} \left[\prod_{i=1}^{p} f(X_i|\theta_i)\right] G^p(\underline{\theta}_{ij}) \qquad (4)$$

where θ_i and X_i are the class and measurement of the ith pixel in the p-array (in any convenient order), $f(X|\theta)$ is the class-conditional density of X, and $G^p(\underline{\theta}_{ij}) = G^p(\theta_1, \theta_2, \ldots, \theta_p)$, which ideally must be known for the type of scene to be classified, but in practice must usually be estimated from an accurately classified sample of the scene or from an analogous scene of known classification.

An experiment was formulated to investigate the extent to which this classifier model can utilize contextual information in satellite-gathered remote sensing data. In order to avoid confounding other effects with the impact of context, it was decided to use a simulated data set generated as follows. A classification of multispectral remote sensing data was selected which had been judged to be very accurate (typically, produced by careful analysis and refinement of multitemporal data). Such a classification could be expected to embody the contextual content of an actual ground scene. Based on the classification map and using the associated statistics of the classes (developed in producing the classification) data vectors were produced by a Gaussian random number generator and composed into a new data set. Thus the new data set had the following characteristics:

1. Each pixel in the simulated data set represented the same class as in the "template"

classification. The template could be considered the "ground truth" for the new data set.

2. All classes in the data set were known and represented.

3. All classes had multivariate Gaussian distributions with statistics typical of those found in real data.

4. All pixels were class-conditionally independent of adjacent pixels.

5. There were no mixture pixels.

Although the simulated data is somewhat of an idealization of "real" remote sensing, its spatial organization is consistent with a real world scene and its overall characteristics are consistent with the context model set out above. In essence, then, what the experimental results based on the simulated data show is the effectiveness of the context classifier given that the underlying assumptions are reasonable. Further experiments are required to generalize the conclusions of these results to real data.

Three data sets were selected to represent a variety of ground cover types and textures. Data set 1 is agricultural (Williston, North Dakota), with ground resolution and spectral bands approximating those of the projected Landsat D Thematic Mapper. Data set 2a is Landsat 1 data from an urban area (Grand Rapids, Michigan). Data set 2b is from the same Landsat frame as 2a, but from a locale having significantly different spatial organization. Each data set is square, 50 pixels on a side.

Figure 2 shows the achieved classification results. The "no context" classification accuracy is plotted coincident with the vertical axis of each graph. Data set 1 was classified using successively 2, 4, 6 and 8 neighboring pixels; data sets 2a and 2b were classified using 2, 4, and 8 neighboring pixels. The results speak for themselves. The accuracy improvement resulting from the use of contextual information is quite significant.

For this experiment, the context distribution $G^p(\theta_{ij})$ was simply tabulated from the "template" classification. But in a real data situation, such a template is not available (else there would be no need to perform any further classification). One can envision a number of ways in which the p-vector distribution might be estimated for a remote sensing application. For example, it could be extracted from a classification of the same area obtained previously. This would require that the area not have changed too greatly in its class make-up since the earlier data were collected and that the earlier classification was reasonably accurate. Or, the distribution might be obtained from a classification of any similarly constituted area. Still another possibility would be to estimate the p-vector distribution for the context classification from a "conventional" classification with "reasonably good" accuracy. All of these methods produce an estimate of the p-vector distribution, and a crucial question on which hinges the utility of this approach is how sensitive the contextual algorithm is likely to be to the "goodness" of the estimate. This question is the subject of ongoing research.

An experiment was formulated to obtain some evidence concerning the feasibility of applying the context classifier to a real data situation. The data set used covered a somewhat larger area of Grand Rapids, Michigan, containing both data sets 2a and 2b. Data from small areas of known ground cover were used to estimate the training class statistics, and data from a disjoint set of areas of known ground cover were used as "test samples" to evaluate the classifier accuracy (unfortunately, the set used for this test was rather small, consisting of only 136 pixels distributed among 4 urban classes).

A non-contextual classification was performed and found, based on the test set, to be 81.6 percent accurate. The p-vector distributions were estimated from this classification and used to perform contextual classifications using four and eight nearest neighbors. The four-neighbor classification was 83.1 percent accurate; the eight-neighbor classification was 84.6 percent accurate. For this case, then, some improvement in classification accuracy was achieved by incorporating context in the decision process, although the improvement was not as dramatic as for the simulated data sets. Whether this is due to poor estimation of the p-vector distributions or simply to less contextual information in the overall data set will be established by further investigation.

III. MULTIPROCESSOR IMPLEMENTATION OF CLASSIFICATION ALGORITHMS

Classification algorithms such as the context classifier (and even much simpler algorithms used for remote sensing data analysis) typically require large amounts of memory and computation. These are said to be processor bound. Since many available systems, such as the IBM 360/370 VM or the PDP 11/70 UNIX, are used on a time-

sharing basis, a large processor-bound program forces the operating system to operate with less overall memory, forcing the memory management to swap large amounts of information in and out of main memory. This reduces the efficiency of processing, forcing the processor to take longer on all jobs involved. For example, when UNIX is under heavy loads (typically three to five processor-bound jobs with 35 to 40 on-line users), the CPU spends up to 60% of its time on operating system tasks such as memory management. One way to speed up the processes would be to add a dedicated special-purpose processor. Through the use of parallel processors, the system throughput could be increased even more. Various dedicated systems have been proposed, such as pipelined processors[6], multimicrocomputer systems[7,8], and special purpose systems[9].

To demonstrate the use of a such a system on a task less complex than the contextual classifier, consider the analysis of Landsat data using a Bayes maximum likelihood classifier (MLC). Landsat measurements are taken from four spectral bands and received as a data vector. Based on decision theory akin to that developed in the previous section the vector is classified by determining the probability that it belongs to each information class and assigning it to the class for which this probability is maximum. In this case, one approach would be to have one processor compute the probability for each of the classes. Such a method of processing would yield a substantial increase in throughput over a dedicated single-processor system.

Consider, for example, the Control Data Corporation (CDC) multiprocessing system consisting of an array of dynamically microprogrammable processors called Flexible Processors (FPs)[6,10,11]. The CDC FP currently has no hardware facilities for floating-point operations, a disadvantage of the system. But the parallelism of the system more than outweighs this fault. The basic clock cycle time is 125 nsec, but since the FP is designed to be connected to as many as 15 other FPs in a parallel and/or pipelined fashion, the effective throughput can be drastically increased, resulting in a potential effective cycle time of less than 10 nsec. The CDC FP has been considered for its use in a large-scale image processing system[12]. Its use in implementing a Bayes maximum likelihood classifier is demonstrated below. The techniques described are to be extended to the contextual classification algorithm.

A configurational diagram of the FP is shown in Figure 3. (This is a preliminary FP design, but the final version should be very similar.) One of the features of the FP is the double-bus architecture which allows the user to manipulate data in 16-bit units. Use of 16-bit integer formats doubles the effective storage capacity of the machine, but 32-bit lengths also are easily handled, which makes it possible to work with the IBM 360/370 floating-point numbers as well as the PDP 11/70 formats. Further, it is possible to implement floating-point operations in software, so the machine is capable of doing floating-point arithmetic as is required by the classification algorithms.

In each FP there are two register files, one called the temporary register file and the other the large register file. Both are divided into 16-bit subunits. If the needed path width is 16 bits, the two files can act like four files, thus creating more addressable user space. A special feature of the temporary file is its separate read and write address registers, which can save much CPU time in many types of matrix operations. It is possible to do either a read or a write to either file and simultaneously increment (or decrement) the address, further increasing throughput. The temporary file is 16 words by 32 bits wide, while the large file is 4096 words by 32 bits wide. All of the register files consist of 60 nsec random access memory (RAM).

There are three general purpose registers (GPRs) called the E, F and G registers. All of these registers are connected to the arithmetic logic unit (ALU). The E and G registers are readable only through the ALU. It is possible to shift the GPRs separately as well as combining the E and F registers to do a double-length shift. The output of the ALU is treated as a register which is accessible in eight-bit units. Separate from the ALU is a hardware integer multiplier, which takes two eight-bit numbers and multiplies them to produce a 16-bit product. The input registers are the P and Q registers, which are each 16 bits wide. The user can choose which of the two groups of eight bits are to be multiplied.

The FP is equipped with four index registers and four corresponding index-compare registers. There are four general compare registers called maintenance compare registers. All of the above are used for looping and can be incremented or decremented during any statement not accessing those registers.

The FP is equipped with a jump stack, so it is capable of handling standard types of program calls such as subroutine jumps. This stack is only 10 bits wide, and is, consequently, not suitable for storing data.

Input/output (I/O) for the FP depends on the overall system (i.e., the FP array and its host machine). Direct I/O among FPs and/or the host is done via the AIO, AI1, AR0 and AR1 registers. There are interlinked memory units on the FP system which are accessed via the z_{in} and z_{out} registers. Interrupts are handled through the Intr (Interrupt) registers, so I/O is fairly easy and very fast (about 32 megabytes per second).

The busses are connected to a register pair called the BRG pair. These are linked to the panel lights on the machine and can be used for breakpointing or as a GPR during execution.

Figure 4 shows how the FPs are linked to the host and to each other. The shift network is one means of inter-FP communication, the other being through interlinked memories. Each FP can address certain memory banks, which can be accessed by certain other FPs. The shared memory (160 nsec cycle time) is especially useful when it is necessary to transfer large amounts of data between FPs.

Figure 5 shows a coding form for the FP which shows, for example, that it is possible to conditionally increment an index register, do a program jump, multiply two eight-bit integers, and do a logical operation on the E and G registers, all simultaneously. This type of operational overlap in conjunction with the use of many processors executing concurrently greatly increases the effective speed of the FP array.

The ability to do a fast matrix multiple is at the heart of efficiently implementing the Bayes maximum likelihood classifier. The form for the matrix multiplications is[1]:

$$(X-U_i)^t \ (C_i^{-1}) \ (X-U_i),$$

where X is the data vector, U_i is the mean vector for the ith class, and C_i is the covariance matrix for the ith class.

Consider the use of the FP array to perform these classifications. Assume there are m distinct classes and the computer system contains p FPs. Each FP is assigned to process m/p classes. The large file in each FP is initialized with the inverse of the covariance matrices and mean vectors for each class it was assigned. The current data vector is stored in each FP in the temporary file. When a new data vector is loaded into an FP it overwrites the previous one. For simplicity, but without loss of generality, in the following assume that m = p. If m is greater than p, then in each FP instead of applying just one inverse covariance matrix to the data set several would be applied. This will, of course, increase the execution time by a factor of approximately m/p.

In standard arithmetic, one would first multiply $(X-U_i)^t$ and C_i^{-1}, creating a new vector. This vector would then be multiplied by $(X-U_i)$ resulting in a scalar. In our implementation, the order has been somewhat altered. $(X-U_i)^t$ is multiplied by a column of C_i^{-1}, accumulating the results in a variable called "sum." After this is done for column j of C_i^{-1}, "sum" is multiplied by $(X-U_i)_j$ (the jth element of $(X-U_i)$), accumulating the result in a variable called "hold" and re-initializing "sum" to 0[2,3]. The following is a "pidgeon ALGOL" description of the process for one pixel:

```
total=0
for j=1 to n do
    begin;
    sum=0;
    for k=1 to n do
        sum=sum+D[k]*C_i^-1[k,j];
    hold=hold+sum*D[j];
    end;
```

n = dimension of covariance matrix
D[k] = kth element of $(X-U_i)$, computed when X is loaded
$C^{-1}[k,j]$ = element in the kth row and jth column of C_i^{-1}

At the end of the routine, the value contained in the "hold" variable is the desired scalar. This algorithm requires fewer stores and fetches than the standard algorithm, so it shortens the run time of the process. All pointers are kept in the index register, further simplifying the process. Finally, because only two accumulators are used, the three GPRs can be kept free for the floating-point operations, while the accumulators are stored elsewhere.

One way to perform this algorithm is to have the host initially send C_i^{-1} and U_i to FP i. The host then sends the current data vector X to FP 0, then to FP 1, FP 2, etc. As soon as the FP receives the data vector, it begins the calculation of the value of the discriminant function. After the host gives all FPs the data for pixel (i, j), it waits until FP 0 has calculated the value for its discriminant function. The host then retrieves the value of the discriminant function and loads FP 0 with the data vector for the next pixel. The host executes this process for all the FPs. When the last FP has transmitted the result, the host does a compare and stores the class index corresponding to the maximum of the discriminant values computed for this pixel. Thus, the compares are done by the host

while the FPs are computing the discriminant functions for the next pixel, minimizing delay.

Allowing 40 FP machine cycles for each floating point addition and 9 FP machine cycles for each floating point multiply[13], the number of machine cycles is as follows:

where j = number of pixels and
n = number of measurements (size of data vector):

setup and clear registers:	9
load mean:	2n
load covariance matrix:	$4n^2$
load and normalize data vector:	$42jn+j$
inner loop of algorithm:	$56jn^2$
outer loop of algorithm:	$61jn$

$$56jn^2 + 103jn + 4n^2 + 2n + j + 9.$$

This assumes that m, the number of classes, equals p, the number of processors. If m is greater than p, the runtime may be approximated by multiplying by $\lceil m/p \rceil$.

Exact comparisons of the FP array with other systems are difficult without detailed information about factors such as pre- or post-processing done by the host machine and the data precision used. However, to give a general idea of the effectiveness of this approach, consider a 256 x 256 classification of Landsat data (n=4) using 16 classes and a complete array of 16 FPs. The total processing time is approximately 10.7 sec. ESL[14] states that their array processor gives up to an increase of 25 times over the IBM 370/158. On the classification of four channels into eight classes, their time is 6.3 sec.

Due to the organization of the FP and the fact the user can microprogram it, accurate mathematical analyses of FP algorithms are complex. In order to study these timing questions, a simulator for a single FP was developed[15]. It has now been expanded to handle multiple FPs. The maximum likelihood classifier is currently being implemented on the simulator to confirm the analytical timing results and to provide a working classifier program written in FP assembly language which could be run on the actual FP hardware. This will allow an accurate cost-effectiveness study.

IV. CONCLUSIONS

The preliminary results from the use of context in classification are promising. By studying ways of estimating the p-vector, choosing the size and shape of neighborhood, etc., it may be possible to develop a highly accurate classifier for context-rich scenes.

The discussion of performing of the maximum likelihood classifier demonstrates one way in which a multiple processor system can be used to speed up the processing of image data. The implementation of the classifier on the simulator and eventually on the actual FP system will provide hard data to verify the effectiveness of this approach.

Through the use of parallel, pipelined, and/or special purpose computer systems, such as the CDC Flexible Processor system, the types of computations required for the context classifier and other computationally demanding processes can be implemented efficiently. This will not only reduce the computation time required to do contextual classification but will as well allow the investigation of techniques which may otherwise be considered infeasible.

V. REFERENCES

1. P. H. Swain and S. M. Davis, eds., Remote Sensing: The Quantitative Approach, McGraw-Hill, New York, 1978.

2. R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural Features for Image Classification," IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-3, pp. 610-621, November 1973.

3. R. L. Kettig and D. A. Landgrebe, "Classification of Multispectral Image Data by Extraction and Classification of Homogeneous Objects," IEEE Trans. Geoscience Electronics, Vol. GE-14, pp. 19-26, January 1976.

4. J. S. Weszka, C. R. Dyer, and A. Rosenfeld, "A Comparative Study of Texture Measures and Terrain Classification," IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-6, pp. 259-285, April 1976.

5. J. R. Welch and K. G. Salter, "A Context Algorithm for Pattern Recognition and Image Interpretation," IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-1, pp. 24-30, January 1971.

6. G. R. Allen, L. O. Bonrud, J. J. Cosgrove, and R. M. Stone, "The Design and Use of Special Purpose Processors for the Machine Processing of Remotely Sensed Data," Conference on Machine Processing of Remotely Sensed Data, IEEE Cat. No. 73CH0834-2GE, pp. 1A-25 to 1A-42, October 1973.

7. H. J. Siegel, "Preliminary Design of a Versatile Parallel Image Processing System," Third Biennial Conference on Computing in Indiana, pp. 11-25, Indiana Univ., Bloomington, IN, April 1978.

8. H. J. Siegel, P. T. Mueller, Jr., and H. E. Smalley, Jr., "Control of a Partitionable Multimicroprocessor System," 1978 International Conference on Parallel Processing, IEEE Cat. No. 78CH1321-9C, pp. 9-17, August 1978.

9. K. S. Fu, "Special Computer Architectures for Pattern Recognition and Image Processing-An Overview," 1978 National Computer Conference, pp. 1003-1013, June 1978.

10. Control Data Corp., "Cyber-Ikon Image Processing System Design Concepts," Digital Systems Division, Control Data Corp., Minneapolis, Minnesota, January 1977.

11. Control Data Corp., "Cyber-Ikon Flexible Processor Programming Textbook," Digital System Division, Control Data Corp., Minneapolis, Minnesota, November 1977.

12. J. L. Kast, P. H. Swain, and T. L. Phillips, "The Feasibility of Using a Cyber-Ikon System as the Nucleus of an Experimental Agricultural Data Center," LARS Contract Report 021678, Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, Indiana, February 1978.

13. K. W. Krause, "Use of the CDC Cyber-Ikon System for a Bayes Maximum Likelihood Classifier," unpublished report, May 1978.

14. ESL, Incorporated, "Advanced Scientific Array Processor," descriptive manual, ESL, Java Drive, Sunnyvale, California.

15. K. W. Krause, "Use of the CDC Cyber-Ikon Simulator," unpublished report, August 1978.

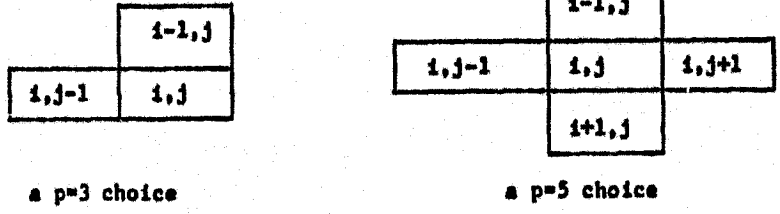
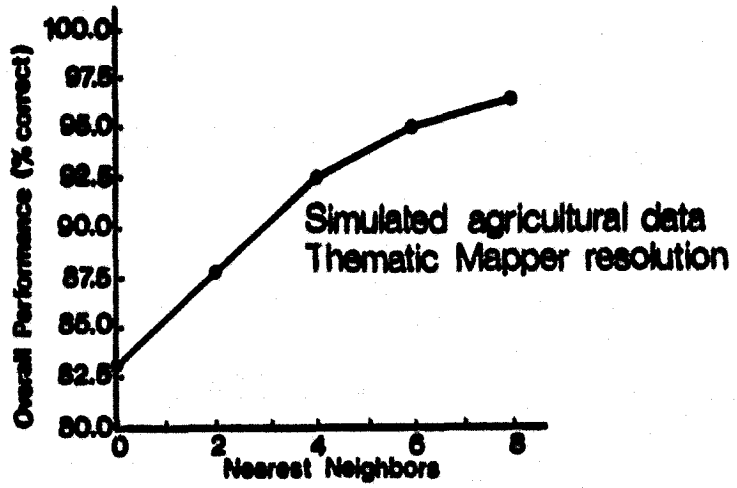
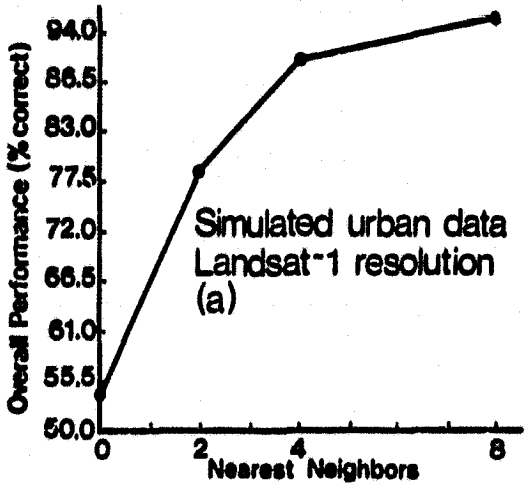


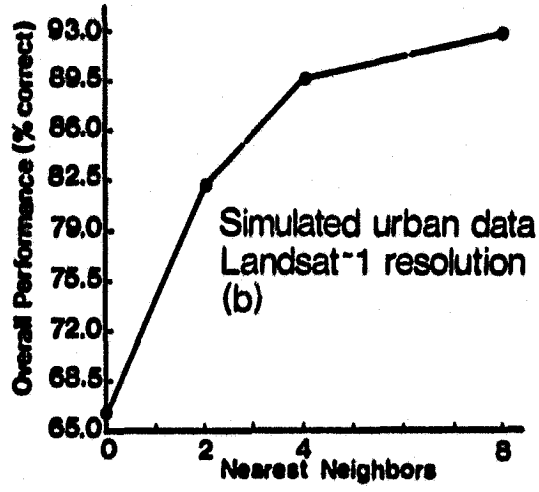
Figure 1. A p-pixel neighborhood.



(a)



(b)



(c)

Figure 2. Results for simulated data: (a) data set 1, (b) data set 2a, (c) data set 2b.

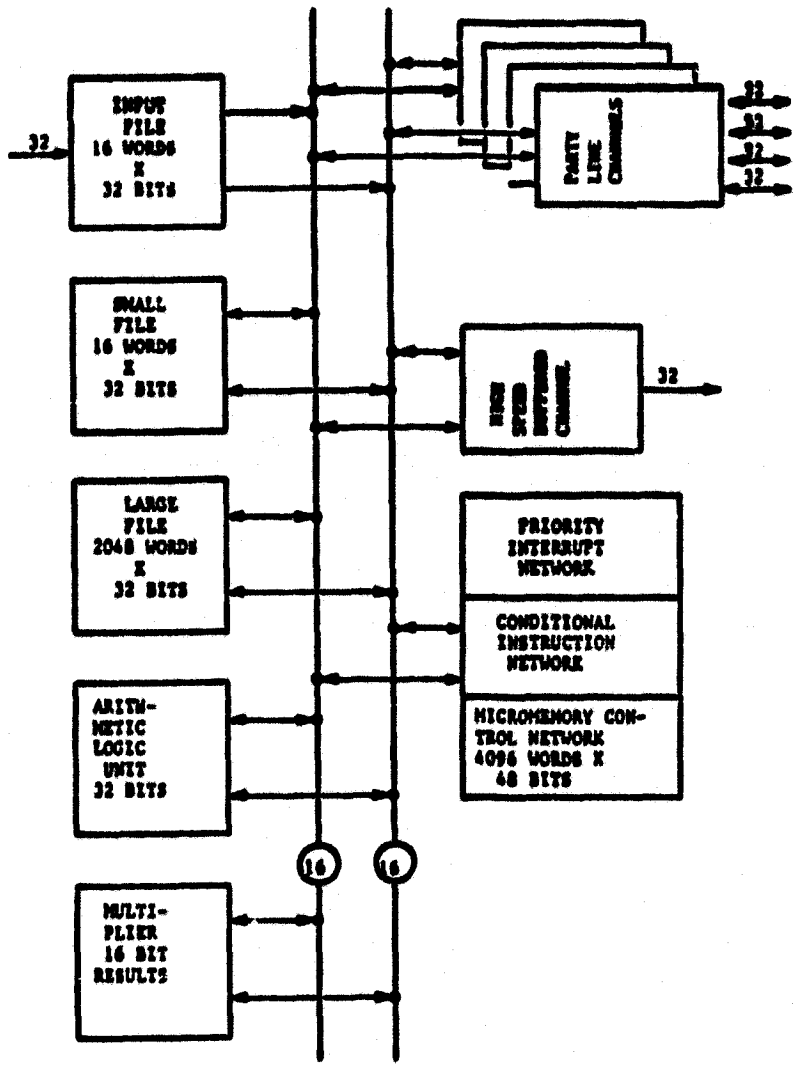


Figure 3. Data path organization in the CDC Flexible Processor[18].

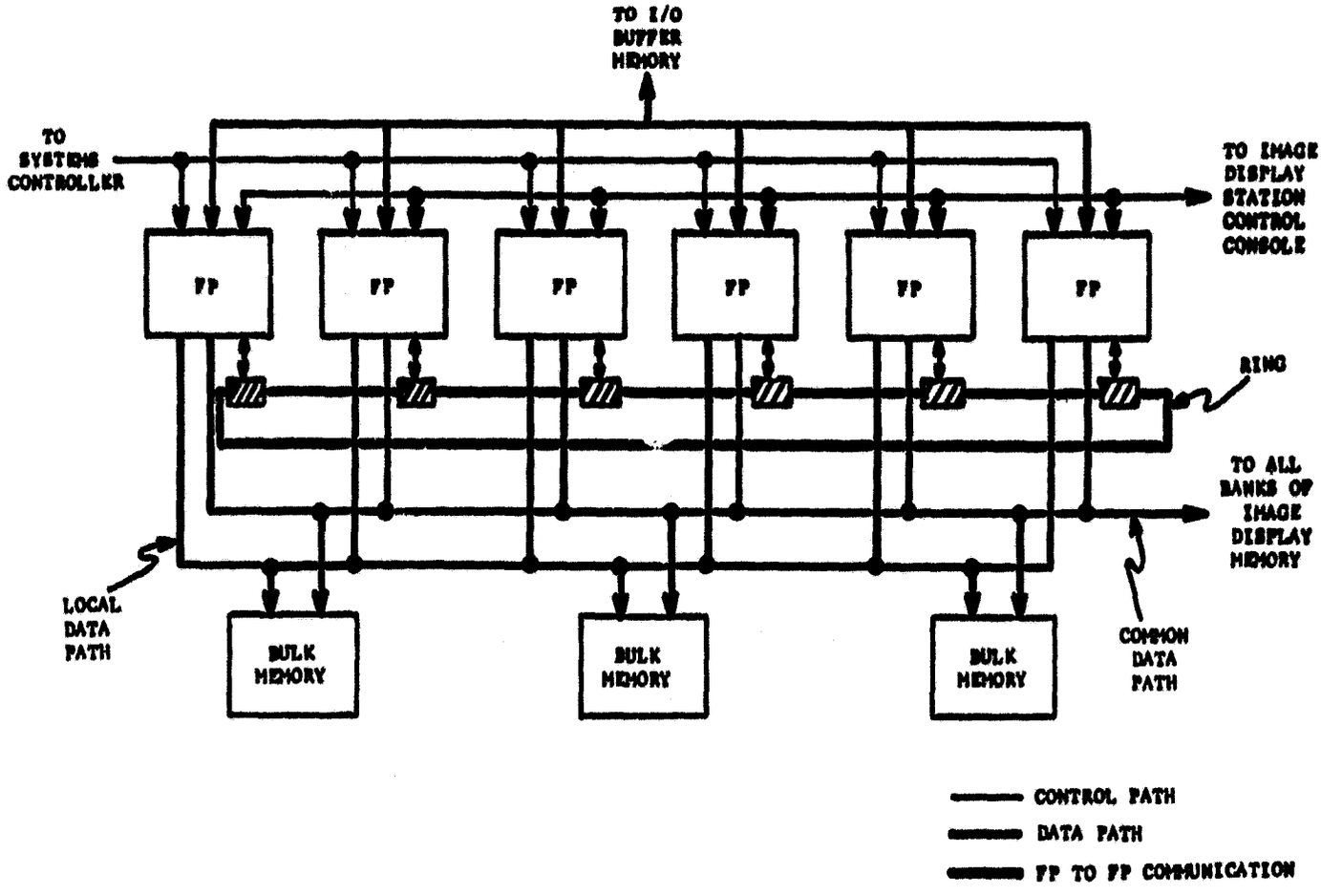


Figure 4. FP array block diagram[10].

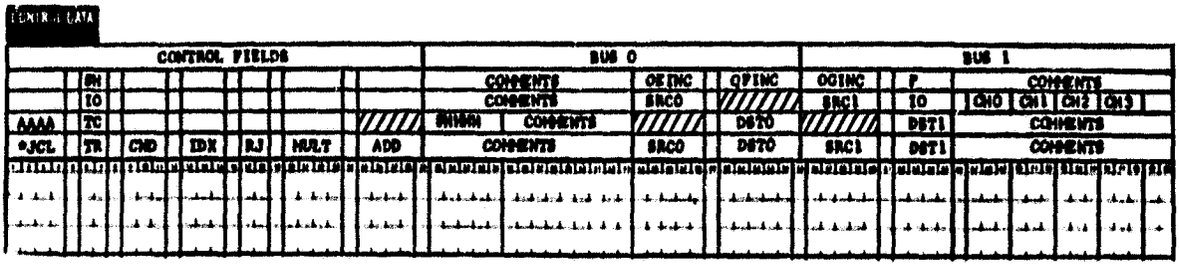


Figure 5. Flexible Processor coding form[11].

Philip H. Swain is assistant professor of electrical engineering, Purdue University, and program leader for Data Processing and Analysis Research at the University's Laboratory for Applications of Remote Sensing (LARS); B.S.E.E., Lehigh University; M.S.E.E. and Ph.D., Purdue University. Prof. Swain has been affiliated with LARS since 1966 and has contributed extensively to the development of data processing methods for the management and analysis of remote sensing data. His areas of specialization include theoretical and applied pattern recognition and methods of artificial intelligence. He is co-editor and contributing author for the textbook Remote Sensing: The Quantitative Approach (McGraw-Hill, 1978).

Howard Jay Siegel is an assistant professor in the School of Electrical Engineering at Purdue University and on the research staff of Purdue's Laboratory for Applications of Remote Sensing (LARS). His research interests include parallel processing, multimicroprocessor systems, speech processing, natural language processing, and image processing. Dr. Siegel received both the S.B. degree in EE and S.B. degree in Management in 1972 from MIT. He received the M.A. and M.S.E. degrees in 1974, and Ph.D. degree in 1977, all from the Department of Electrical Engineering and Computer Science at Princeton University. He is a member of the Eta Kappa Nu and Sigma Xi honorary societies.

Bradley W. Smith received the B.S. degree in December 1978 from the Electrical Engineering School at Purdue University. He is currently pursuing the M.S.E.E. degree at Purdue and is on the research staff at LARS. He is a member of the engineering honorary Tau Beta Pi and the science honorary Phi Kappa Phi.